MITOCW | ocw-6.033-lec9

0:00:00 So, before we begin, I just want to make sure you 0:00:03.245 guys all remember about the quiz on Friday. 0:00:06.086 So the quiz is going to be in Walker at 2 pm. 0:00:09.061 Everybody goes to Walker. The quiz is open book. 0:00:12.239 You can bring your own notes. You can bring a class notes, 0:00:16.094 bring the readings from class; you can bring your calculator. 0:00:20.151 But please don't bring any computers, phones, 0:00:23.127 PDAs, things like that. So make sure you have all your 0:00:26.711 stuff printed out before you show up if you've been taking 0:00:30.565 notes on a laptop. OK, so what we're going to do 0:00:35.761 today is just quickly, well, we're going to do an 0:00:40.333 introduction to networking. But quickly I just want to 0:00:45.38 finish the topic of caching that we began, that sort of 0:00:50.523 introduced at the very end of the lecture last time. 0:00:55.38 So, if you guys remember, we were talking about our 0:01:00.142 pipeline WebServer system that consisted of these three stages: 0:01:06.047 a networking stage connected up to our HTML stage connected up 0:01:11.857 to our disk stage. OK, and we did a little 0:01:16.547 performance analysis of the system. 0:01:18.834 We looked at what the cost of each of these stages is, 0:01:22.4 what sort of throughput of each one of these stages is. 0:01:26.033 We said that as long as we split this networking module up 0:01:29.868 into ten submodules, we can get the throughput of 0:01:33.097 this thing to be about 100 requests per second. 0:01:37 We said the latency of this was 1 ms. 0:01:39.219 So that means it can process a thousand requests per second. 0:01:42.857 We said the latency of the disk was 10 ms. 0:01:45.386 So that means it can also process 100 requests per second. 0:01:48.9 And we said, remember, that when we are 0:01:51.243 looking at a pipeline, the sort of throughput of the 0:01:54.388 entire pipeline is going to be bottlenecked by the slowest 0:01:57.903 stage in the system. And so if we look at this, 0:02:01.846 we see that these two stages are both running at 100 requests 0:02:06.108 per second. But we have a very simple way 0:02:08.949 of making this first network stage be able to process more 0:02:12.998 requests per second, right, because we simply can 0:02:16.407 replicate the number of threads that are sending data out over 0:02:20.74 the network. So, for example, 0:02:22.728 if we went from ten nodes here to 100 nodes, 0:02:25.783 we could increase the throughput of this stage to 0:02:29.192 1,000 requests per second, which means that now the 0:02:32.743 bottleneck stage that we have is simply the disk stage. 0:02:38 So the question is, is there something we can do to 0:02:40.81 increase the throughput of the disk stage? 0:02:43.114 If you think about it, at first it may seem like, 0:02:45.812 well, there's probably no way that we can do anything because 0:02:49.185 the disk takes 10 ms, you know, every page takes 10 0:02:51.995 ms to read in. So what are we going to do 0:02:54.243 about that? And there's a very sort of 0:02:56.323 standard answer to that that you guys have all seen before, 0:02:59.583 and that answer is caching. OK so the simple idea is that 0:03:04.127 we're going to take this IO stage, or this disk stage, 0:03:08.016 with this disk that runs at 10 ms. 0:03:10.438 What I've shown here is a very simple piece of pseudo code that 0:03:14.987 might correspond to what this sort of read or the get page ID 0:03:19.389 function for this IO stage does. It simply calls some read 0:03:23.572 function that reads page ID off the disk and then returns the 0:03:27.974 page. If we add a cache to this 0:03:30.176 system, suppose we have an in memory cache that can retrieve a 0:03:34.651 page in 0.1 ms. In that case, 0:03:37.584 the way that we can use that cache is just before every time 0:03:40.235 we go to the disk, we can check and see if the 0:03:42.258 page that we're trying to load is in the cache. 0:03:44.325 And then only if it's not in the cache do we need to actually 0:03:47.022 go to the disks. And we get a miss on the cache. 0:03:49.134 We go check the disk to see if the page is available. 0:03:51.471 So we can extend the code in a very simple way to take 0:03:53.853 advantage of this. We simply say, 0:03:55.292 we look up in the cache first, and then if the page is null or 0:03:58.033 empty or whatever, we can't find it in the cache, 0:04:00.191 then we go ahead and look it up on the disk. 0:04:02.123 And then we add that result to the cache. 0:04:05 So there's a couple of little details that we need to work 0:04:08.772 through. But let's first look at what 0:04:11.154 the performance of this system is going to be, 0:04:14.132 or how this is going to impact the performance of what we've 0:04:18.036 been doing. So if we come over here, 0:04:20.352 if we think about what the cost of accessing a page on this 0:04:24.191 system would be, well, we're always going to 0:04:27.036 have to check the cache, right? 0:04:29.022 That's the first thing we do. So I'm going to write C for 0:04:34.026 cost of accessing the cache, where cost is in this case 0:04:38.08 going to be expressed in something like the number of 0:04:41.983 milliseconds to do a lookup in the cache, and then plus the 0:04:46.337 cost of looking it up on the disk. 0:04:48.815 But we only have to look up on the disk some of the time, 0:04:53.018 right, so there is some probability of getting a miss 0:04:56.922 times the cost of that miss. OK, so in this case, 0:05:01.559 we said the cost of going to the cache was 0.1 ms. 0:05:05.379 The cost of going to the disk is 10 ms. 0:05:08.341 OK, and now what we're left is to figure out what the 0:05:12.395 probability of this miss is. So if you think about a system 0:05:16.916 for a little bit, you might say, 0:05:19.333 well, OK, what does the system look like? 0:05:22.451 What's the probability that we are actually going to get a hit 0:05:27.206 or miss? And if what the applications 0:05:30.395 are doing is simply generating random requests for random 0:05:33.186 pages, which might be one model of how the application behaves, 0:05:36.276 then it doesn't seem like cache is going to buy us very much, 0:05:39.266 right, because the cache is going to be a small fraction of 0:05:42.157 the total size of the disk. It might be in RAM, 0:05:44.45 so we might have 100 MB of cache, and we might have 10 GB 0:05:47.241 of disk, right? So that means there is a factor 0:05:49.533 of 100 difference between these two things. 0:05:51.627 So if the page requests are random, the probability that 0:05:54.368 something is going to be in the cache is going to be very low. 0:05:57.408 It's going to be only, say, 1%. 0:06:00 But it turns out that almost all applications have an 0:06:03.199 interesting property which is commonly referred to as locality 0:06:06.953 of reference. 0:06:08 0:06:13 What this means is that if you look at which pages a program is 0:06:16.559 likely to access over time, be these pages of data or parts 0:06:19.889 of a program, typically a page that has been 0:06:22.358 accessed recently is very likely to be accessed again. 0:06:25.401 So a simple example might be that if you look at the files 0:06:28.674 that are being used in a computer system, 0:06:30.971 usually there is a small set of files that a user

is working 0:06:34.358 with at any given point in time. You're running certain 0:06:38.195 programs; you're editing certain documents. 0:06:40.477 And there's a huge array of other programs and files and 0:06:43.465 documents that are on the system that you aren't accessing. 0:06:46.616 And when those active files are being accessed, 0:06:49.115 there's a much higher probability of getting a hit of 0:06:51.94 looking at those active files then there is of going to any 0:06:55.091 one of the other files. So even when the difference 0:06:57.808 between these things is, say, a factor of 100, 0:07:00.252 in the case of a Web server, it might be very likely that we 0:07:03.458 would have 90% maybe of the pages that have been accessed 0:07:06.5 are already in the cache. So in that case, 0:07:10.438 sorry, this probability should be 0.9, 90%. 0:07:14.215 OK, so suppose that the probability, sorry, 0:07:17.992 the probability of a hit is 90%. 0:07:20.78 The probability of a miss is then 10%. 0:07:24.107 OK, so if you look at now what this sort of formula evaluates 0:07:29.503 to, we see it's 0.1 plus ten times 0.1. 0:07:34 So, it's 1.1 milliseconds on average, assuming that we get a 0:07:39.073 90% hit rate on our cache. OK, so if you now come back to 0:07:43.889 this diagram, that means that the throughput 0:07:47.587 of this box is one over 1.1, remember, because throughput is 0:07:52.574 just one over latency when we have sort of just this one 0:07:57.304 module, which means that now we can process something like, 0:08:02.292 so this is something like, sorry, this should be one over 0:08:07.108 latency. But this is in milliseconds. 0:08:11.37 So this is 0.11, one, sorry, the number of 0:08:14.758 seconds is 0.0011, right? 0:08:16.79 So, one over 0.0011 is about 900. 0:08:19.5 OK, so we can get about 900 requests per second that we can 0:08:24.411 process instead of just 100. This is approximately equal. 0:08:30 OK, so what we've managed to do is increase the throughput of 0:08:33.19 this stage by about a factor of nine. 0:08:35.104 I mean, now you can see that sort of all three of these 0:08:37.976 stages are close to about 1,000 requests a second. 0:08:40.581 We have increased the performance of the system pretty 0:08:43.4 dramatically by introducing the cache. 0:08:45.367 And you have to remember that this cache is only going to be a 0:08:48.611 good idea when we are sure that we have this locality of 0:08:51.536 reference property. OK, so if the Web server is 0:08:53.982 going to a random page, if the Web server is sort of 0:08:56.694 asked to fetch a random page on every request, 0:08:59.087 the cache is not going to be a good choice. 0:09:01.32 But most of the time Web servers to have this locality 0:09:04.138 property. So the last little detail that 0:09:08.968 we need to talk about when we talk about caches, 0:09:13.618 is the sort of question about how we deal with page eviction. 0:09:19.554 So in this diagram here, when we call add to the cache, 0:09:24.897 if the cache is already full of results, we have to pick 0:09:30.339 something to replace, right? 0:09:34 So we need what's called a page removal or page replacement 0:09:39.164 policy. OK, so you guys presumably have 0:09:42.547 seen different page removal or page replacement properties in 0:09:47.89 6.004 before. I'll just talk about two very 0:09:51.63 quickly: a FIFO policy and a LRU policy. 0:09:55.102 OK, so what FIFO means is First In First Out. 0:10:00 So it says the thing to throw out is the first thing that we 0:10:04.45 loaded into the cache. OK, so if I access a set of 0:10:08.145 pages, suppose I have a three element cache and I access a set 0:10:12.746 of pages, one, two, three, two, 0:10:15.009 one, four, what I'm going to do is I'm going to load this. 0:10:19.309 I'm going to access these three pages. 0:10:22.099 Page 1 will be the first one that's in my cache. 0:10:25.644 So when I try and load page 4, I'm going to evict page 1. 0:10:31 OK, that's what's going to happen in a FIFO system because 0:10:34.239 one was the first one that was loaded. 0:10:36.342 But a least recently used approach says instead of 0:10:39.127 evicting the first thing that was put into the cache, 0:10:42.082 we want to evict the last thing that was read from the cache. 0:10:45.492 So in this case, if I do one, 0:10:47.083 two, three, two, one, and then I get four, 0:10:49.414 the last thing that was read was three, and that's what I'll 0:10:52.767 evict from the cache. So, if you think about the 0:10:55.438 intuition behind these two policies, you sort of realized 0:10:58.621 that this FIFO approach has a problem, which is that it's sort 0:11:02.087 of contrary to this notion of locality of reference, 0:11:04.986 right, because it doesn't capture anything about how 0:11:07.884 frequently a data item is accessed or when it was last 0:11:10.897 accessed. It throws the sort of first 0:11:14.757 access thing out of the cache even if that first access thing 0:11:18.662 is read all the time. LRU sort of captures the 0:11:21.591 intuition which we want, which is that something that's 0:11:25.106 been accessed recently is likely to be accessed again. 0:11:28.556 And something that hasn't been accessed very recently is less 0:11:32.461 likely to be accessed again. So it sort of captures our 0:11:36.64 intuitive idea that we want locality of reference. 0:11:39.731 So the text talks much more carefully about these different 0:11:43.39 page removal policies. But anytime you see anybody 0:11:46.481 talk about a cache, you should sort of immediately 0:11:49.572 think to ask the question, well, what's the page removal 0:11:53.042 policy that's being used? OK, so that basically does it 0:11:56.448 for our discussion of caching and of performance that we 0:11:59.918 started last time. And what we're going to do now 0:12:04.658 is move on to the topic of networking, OK? 0:12:08.291 So -- 0:12:09 0:12:25 So a computer network, you guys should all be familiar 0:12:28.71 of what the idea of a computer network is. 0:12:31.581 It's some sort of a connection that connects multiple computers 0:12:35.922 together. So why are we interested in 0:12:38.443 studying a computer network? What is it that's relevant 0:12:42.224 about a computer network for the purposes of this class? 0:12:46.075 And there's really sort of two primary reasons. 0:12:49.296 The first one is that computer networks are commonly used as 0:12:53.427 components of systems. OK, so anytime we are building 0:12:57.068 a big computer system, it's very likely it's going to 0:13:00.709 involve a computer network. So it's going to be important 0:13:05.639 for us to understand what the properties of computer networks 0:13:09.031 are, and how computer networks affect the design of our system 0:13:12.48 if we want to use them effectively in our systems. 0:13:15.307 And computer networks have a number of uses in the computer 0:13:18.586 systems. They do things like allow us to 0:13:20.791 overcome geographic limits. So what you guys are all 0:13:23.674 familiar with the ability of the Internet to allow you to check 0:13:27.18 the score of some sporting game that's happening on the other 0:13:30.572 side of the country, or to be able to send an e-mail 0:13:33.455 to your parents back home. So obviously, 0:13:36.911 being able to sort of transmit data over long distances is 0:13:40.905 clearly a good thing for many kinds of systems. 0:13:44.129 They also allow us to access remote data. 0:13:46.932 And this is related to the other thing, to the other one. 0:13:50.857 But if you have data

that's not on your computer that you want 0:13:55.132 to get a hold of, you may want to contact your 0:13:58.285 bank and ask them for your bank balance. 0:14:02 And then finally they can be used to physically separate a 0:14:05.995 client and a server. So we talked about the last few 0:14:09.57 lectures, we spent a while talking about the fact that we 0:14:13.495 can use things like threads and address spaces in order to 0:14:17.49 provide a sort of enforced modularity between modules 0:14:21.135 running on the same computer. But there are lots of 0:14:24.64 situations in which we want to actually physically separate the 0:14:28.985 client and the server, right? 0:14:32 Your bank doesn't really want you to be running your copy of 0:14:35.392 Quicken on their server, right, because this notion of 0:14:38.439 enforced modularity we talked about isn't a perfect separation 0:14:41.946 between the client and the server, whereas putting these 0:14:45.108 things on really separate machines that are only connected 0:14:48.386 by this network that is controlled by this piece of 0:14:51.26 hardware that's this network card that sort of talks with the 0:14:54.71 network, that's a good reason may be to separate these things 0:14:58.16 from each other. And then finally, 0:15:01.056 the second major reason why we want to study networks is simply 0:15:04.909 that they themselves are an interesting computer system, 0:15:08.327 OK? So we talked in the first 0:15:10.067 lecture about some of the interesting sort of properties 0:15:13.485 that big computer systems have. Well, there's not many computer 0:15:17.338 systems that are bigger than the Internet, right? 0:15:20.322 It's a perfect example of the giant complex interesting system 0:15:24.112 with all sorts of complicated behaviors. 0:15:26.536 And we'll talk about some of those complicated behaviors 0:15:29.954 today as we overview networks. So the goal of our discussion 0:15:35.636 of networking in some sense is going to be to develop a set of 0:15:41.181 tools that allow us to have universal communication between 0:15:46.454 a number of different clients. OK. 0:15:50 0:15:57 So what do I mean by that? Suppose I have some collection 0:16:00.906 of machines, which I'll just draw as little boxes that are 0:16:04.883 scattered around. And they are connected 0:16:07.604 together. What we want to do is to 0:16:09.906 connect these guys together, OK? 0:16:12.069 And the network system that we are going to design is going to 0:16:16.325 be the interconnect that allows these got to talk to each other. 0:16:20.72 And what we want this network to be able to provide at a very 0:16:24.906 high level is the ability to do any-to-any communication. 0:16:30 OK, so if this is A and this is D, we want to A to be able to 0:16:33.296 send a message to D or anybody else who's in this network. 0:16:36.428 And that should be true for any of the pairs that we can find. 0:16:39.78 And so, in some sense, what we're going to do over the 0:16:42.692 course of the next couple of weeks is sort of see how we 0:16:45.714 design this cloud that sits in between all these different 0:16:48.846 computers. If you like, 0:16:50.054 you can sort of think of this cloud. 0:16:51.978 Often times, the Internet is represented as 0:16:54.285 a cloud. It's just some black box that 0:16:56.318 you sort of send messages into with an address. 0:17:00 And on the other side, the message pops out at the 0:17:03.982 other end. So, we're going to dive into 0:17:07.071 this cloud and see what happens inside of it. 0:17:10.647 So, in order to start to understand some of the 0:17:14.386 challenges of building this cloud, we are going to start 0:17:18.857 looking top down. We're going to look at the 0:17:22.352 biggest issues that we have to deal with, and then we're going 0:17:27.31 to break up those issues into some smaller pieces. 0:17:31.293 So -- 0:17:33 0:17:39 OK, so what are the issues that sort is immediately pop out when 0:17:42.997 you first start thinking about the properties of a network? 0:17:46.678 Well, so there's some technological issues which have 0:17:49.978 to do with things like the physical medium over which 0:17:53.277 you're transmitting messages, or the properties of the links 0:17:57.021 over which you're transmitting them, the rates at which you can 0:18:00.956 send data, the lost rate, the percentage of messages that 0:18:04.509 get lost by the link as it's transmitted. 0:18:08 So those are the kinds of things we mean by technological 0:18:11.674 concerns. And so one kind of 0:18:13.446 technological concern is this, is sort of the different kinds 0:18:17.383 of technology that we might use for transmission. 0:18:20.532 And we saw this notion of D technology over DT before. 0:18:24.01 It's just this idea that technology, just like the 0:18:27.225 technology in your computer system has been evolving 0:18:30.572 dramatically over time. The technology of networks has 0:18:34.858 been evolving dramatically. And we're going to sort of 0:18:38.141 study some of the range of different technologies that we 0:18:41.61 have to deal with. There also are a set of limits, 0:18:44.646 technological, physical, fundamental limits 0:18:47.247 that are associated with networks. 0:18:49.292 So, these are things like the speed of light, 0:18:52.017 OK? We simply can't send messages 0:18:54 faster than the speed of light, right? 0:18:56.292 These are messages that are being transmitted down a wire. 0:19:01 And they propagate at some speed. 0:19:02.758 And that's going to be fundamentally a bottleneck in 0:19:05.562 the design of a computer system. If I have to send a message 0:19:08.804 from here to California, that message isn't going to get 0:19:11.828 there any sooner than the amount of time it takes for light to 0:19:15.18 travel across the country. And the amount of time that it 0:19:18.258 takes to transmit a message across the country is nontrivial 0:19:21.501 in computer time. It might be a couple hundred 0:19:23.975 milliseconds. And that's a long time as we 0:19:26.228 saw before it when we have a processor that can execute a 0:19:29.251 billion instructions a second. OK, so the other interesting 0:19:33.806 issue that's going to come out about networks is that networks 0:19:37.741 are a shared infrastructure, OK? 0:19:40 0:19:45 What that means is that there are multiple users who are 0:19:48.179 simultaneously using a network. You guys, of course, 0:19:51.127 all know this from using the Internet. 0:19:53.265 And all the networks that we are going to study in this class are 0:19:56.213 basically a shared infrastructure. 0:19:58.121 And we'll talk in a minute about why the networks are 0:20:01.127 fundamentally almost always going to be sort of shared. 0:20:05 They're going to have to transmit data for multiple 0:20:07.812 different users at the same time. 0:20:09.612 And that's going to present a number of challenges about how 0:20:12.931 we ensure that the network is fair, that everybody gets to 0:20:16.137 send their data when they want to send data, 0:20:18.556 how we sort of allow multiple people to access the same 0:20:21.593 physical wire at the same point in time. 0:20:23.787 OK, so what I want to do is I'll talk about these two things 0:20:27.106 now in order. I'm going to start off by 0:20:29.243 talking about technology. So -- 0:20:32 0:20:52 So the first interesting technological problem, 0:20:54.994 which I've already mentioned is that these networks are 0:20:58.509 extremely diverse. They are

heterogeneous. 0:21:01.179 OK, networking technology has evolved a great deal in the past 0:21:05.15 20 years or 30 years since networks first started being 0:21:08.665 designed. And that means there is a huge 0:21:11.204 sort of range of devices that we might have to transmit data 0:21:15.045 over. So let me show you what I mean. 0:21:17.388 This is just a graph showing the rate of transmission, 0:21:20.839 the sort of number of bits per second that you can send over 0:21:24.679 different networking protocols. And the thing to take away from 0:21:30.457 this first is to note that on the Y axis here, 0:21:34.024 this is an exponential scale. So the technology at the very 0:21:38.621 bottom, things like early telephone modems, 0:21:41.951 could maybe send data at 10 kb per second, whereas these sort 0:21:46.707 of high-end router class devices that are running the Internet or 0:21:51.78 sort of very fast desktop Ethernet kinds of connections 0:21:56.06 that are coming out today can send more like, 0:21:59.548 say, in this case we have 10 Tb a second, right, 0:22:03.274 so one times ten to the tenth. So we are talking about a 0:22:08.829 factor of ten to the seventh or ten to the eighth difference in 0:22:13.365 performance in terms of the number of bits that these 0:22:17.17 different kinds of devices can actually transmit. 0:22:20.682 So, that's pretty dramatic. And that's going to make it 0:22:24.634 difficult to design computer systems. 0:22:28 OK, and these kinds of variations occur not only at the 0:22:32.853 sort of number of bits that we can send per second, 0:22:37.348 but also in terms of the propagation delay of the links, 0:22:42.292 OK? So if I have a local area 0:22:44.808 network that's connecting two computers in my office together, 0:22:50.202 I may be able to transmit a message from one computer to the 0:22:55.505 other in an order of microseconds, 0:22:58.471 OK? But as we said before, 0:23:01.562 to send a message all the way across the country might take 0:23:05.683 200 ms. To send a message across the 0:23:08.169 Pacific Ocean by way of the satellite in orbit around the United 0:23:12.005 States might take a second. To send a message to Mars might 0:23:16.125 take tens of seconds or a minute. 0:23:18.398 OK, so there's this huge range in terms of transmission times 0:23:22.661 that some of these different links have. 0:23:25.431 So, fundamentally these two things, bit rate and propagation 0:23:29.622 delay are going to tell us how long we would expect it to take 0:23:33.956 for a message to travel from one place to the other. 0:23:39 So for example, the amount of time, 0:23:42.371 so we can think of a communication link, 0:23:46.239 if you like, as being a piece of pipe, 0:23:49.909 which has some width that is the bit rate. 0:23:53.975 OK, so the width of the pipe is the number of bits that I can 0:23:59.925 cram down this thing. So, a wider pipe I can shove 0:24:04.639 more data down it. But no matter how much data I 0:24:07.391 can shove down it per second, there still is some link to 0:24:10.671 this pipe, right? And that's the sort of 0:24:12.954 separation between the sender and receiver. 0:24:15.414 And that's bottlenecked, as we said before, 0:24:17.873 by the speed of light. And that's going to affect how 0:24:20.918 long it takes for a message to get from, say, 0:24:23.495 one endpoint to the other. So if you think about the total 0:24:26.833 time to transmit a message from one guy to the other, 0:24:29.878 it's going to be the amount of time it takes to pack the bits 0:24:33.391 into the beginning of the pipe, right, and then the amount of 0:24:36.905 time that it takes for all those bits to come out the other end. 0:24:42 So it's going to be some combination of the propagation 0:24:48.272 delay plus the transmission delay. 0:24:52.106 The propagation delay is just going to be equal to the length 0:24:58.611 divided by, say, for example, 0:25:01.863 the speed of light. And we're going to need to add 0:25:08.218 to that the transmission delay, which is simply going to be the 0:25:14.871 number of bits we have to send divided by the bit rate, 0:25:20.665 the number of bits per second. OK, and so this is the time to 0:25:27.103 send along one link. OK, so it's worth remembering 0:25:31.757 that this time to send equation, this is only to send 0:25:35.03 along a single link. If we have to send more data, 0:25:38 if there are multiple people, for example, 0:25:40.484 who are waiting to get access to a link, that may increase the 0:25:44.181 delay that it takes to send a message. 0:25:46.424 Or if we have to relay a message over multiple hops and 0:25:49.696 there are some intermediate node in between each of those hops, 0:25:53.454 then that sort of processing at each intermediate node may take 0:25:57.212 some additional time as well. OK, so if we look at a graph of 0:26:02.081 some of the properties of networks, it's interesting to 0:26:05.706 sort of look at the rate of change of these different 0:26:09.196 technologies. So this is just showing what 0:26:11.949 Moore's Law is. So we saw this before. 0:26:14.432 Processor speed doubles every 18 months according to Moore's 0:26:18.393 Law. If you look at the size of the 0:26:20.675 Internet, so the number of things that we have to connect 0:26:24.435 together, well, that's been doubling every 13 0:26:27.388 months. So this has this sort of 0:26:30.309 exponential behavior that Moore's Law has as well. 0:26:32.981 So we are talking about sort of the number of things that we 0:26:36.2 have to connect together is just growing at this massive rate. 0:26:39.527 And that's a real challenge for designing the kinds of protocols 0:26:42.963 that we are going to need to use in these networks. 0:26:45.69 The other thing that we see that's been going up at the same 0:26:48.909 rate or even faster is the amount of traffic that's being 0:26:51.963 sent over the Internet. So more and more people are 0:26:54.69 sending data. And again, this affects the 0:26:56.872 sort of, this is the challenge for designing protocols that are 0:27:00.254 able to scale up and up and up to these greater and greater 0:27:03.418 rates. The largest link capacity in 0:27:06.847 the past few years has actually started to go up as fast as 0:27:10.675 double once every seven months. So this is the size of the 0:27:14.438 largest pipe that's on the Internet, say, 0:27:17.077 connecting lots of these big service providers on the 0:27:20.509 Internet together. And this is just been getting 0:27:23.612 larger and larger and larger as well. 0:27:25.988 And so that's, again, means in order to 0:27:28.496 exploit all this additional link capacity, we need faster 0:27:32.192 computers. And the rate of change of 0:27:34.501 technology is just extremely high. 0:27:38 Interestingly enough, of course, as we said before, 0:27:40.958 the speed of light never changes. 0:27:42.852 So this is a fundamental constraint that we're always 0:27:45.928 going to have to deal with when we are building these things. 0:27:49.479 But it's also the case that if you look at, there's another 0:27:52.852 number that you can look at. And in the context of the 0:27:55.988 Internet, this number is, is bits per second per 0:27:59.065 dollar. OK, so what does that mean? 0:28:01.076 This is the amount of money that it costs to, 0:28:03.68 for example, send one bit per second over 0:28:06.047 the Internet. And this number just has not 0:28:10.153 been scaling up very fast. If you look at this, 0:28:13.692 it's a factor of ten cheaper today than it was in 1980 to 0:28:18

scaling up very fast. If you look at this, [illegible] send the same amount of data per second out over the Internet. 0:28:22.692 So, why is that? That seems strange, 0:28:25.384 right? I mean, if you look at the 0:28:27.846 fundamental technology, the technology itself has 0:28:31.538 gotten much faster. If you look at the Ethernet 0:28:35.465 network that you might have in your home or your office, 0:28:38.449 this thing has gotten, in 1980 you might have had one 0:28:41.271 megabit a second that you could transmit over this Ethernet. 0:28:44.472 You can now, for $100 go to Best Buy and buy 0:28:46.806 something that can transmit a gigabit a second over an 0:28:49.682 Ethernet. OK, so we are talking about a 0:28:51.744 huge scale up in technology in terms of the performance of the 0:28:55.054 networks. The issue here is that there is 0:28:57.224 a human cost that's associated with sending data out over the 0:29:00.48 Internet. And if I want to send data from 0:29:03.736 here to California, there's a physical wire that 0:29:06.369 that data has to transfer over, right? 0:29:08.441 And somebody owns that wire. And it cost somebody a lot of 0:29:11.634 money to cut the holes in the ground where that wire runs and 0:29:14.995 to knock holes through buildings and to get rights from all the 0:29:18.468 cities that that wire runs through to dig holes in their 0:29:21.549 roads, right? So, there is some very large 0:29:23.846 human cost associated with setting up these sort of very 0:29:26.927 wide area internets. And you see the same kind of 0:29:30.605 thing not just in wire but in wireless technology with 0:29:33.643 wireless companies paying huge amounts of money for access to 0:29:37.082 the rights to use a particular part of the wireless spectrum. 0:29:40.522 So this is these bits per second per dollar is growing 0:29:43.56 slowly, OK, and the reason for that is human costs. 0:29:47 0:29:52 OK, and what this argues for, what this suggests that we are 0:29:55.224 going to need to do when we are building these internets or 0:29:58.393 building these large networks is to share these network links. 0:30:02 So, if every time we added a new host to the network, 0:30:05.556 we had to run a wire that connected it to every other host 0:30:09.455 in the network. Obviously, that would be hugely 0:30:12.601 expensive, so we want to avoid that. 0:30:14.995 And so, we are going to end up because of these costs of 0:30:18.757 physically running the wires having to share network 0:30:22.245 connections. And that brings us to sort of 0:30:25.049 the next topic that I want to talk about, which is this topic 0:30:29.153 of sharing. 0:30:31 0:30:40 OK, so just that everybody is clear, this notion of having 0:30:44.621 pair-wise links, a connection from every host to 0:30:48.189 every other host is just not practical. 0:30:51.27 So suppose I have them set of nodes like this, 0:30:54.918 if I want to connect all of them together by their own set 0:30:59.54 of independent wires, I don't know if I got all of 0:31:03.513 them there, but then the cost is going to be, if I have N hosts, 0:31:08.621 the cost is going to be N squared wires. 0:31:13 OK, and that is just too expensive, right? 0:31:15.897 If you have a million hosts, you're going to have a trillion 0:31:20.067 wires. So we don't want to be building 0:31:22.681 anything that has a trillion components in it. 0:31:25.862 It's going to be really, really expensive to deploy. 0:31:29.466 So what we end up doing is multiplexing. 0:31:33 OK, so what do I mean by that? I mean this notion of sharing 0:31:38.466 links. So let's look at a very 0:31:41.153 simplified kind of an architecture for something like 0:31:45.971 the Internet. So suppose I have two hosts, 0:31:49.769 two nodes, two computers, A and B that I want to be able 0:31:54.865 to communicate over a network to some other pair of nodes, 0:32:00.146 C and D. OK, well in a multiplex 0:32:03.564 network, what we're going to do is we're going to have a set of 0:32:07.782 these boxes that are going to connect these nodes together. 0:32:12 0:32:19 So I just sort of shown one possible network topology. 0:32:22.105 But what these boxes are, they are sometimes called 0:32:25.035 switches. And they are the connection 0:32:27.145 points that are scattered in various locations around the 0:32:30.426 Internet that transfer data from one location to the other. 0:32:33.825 So in your home, you may well have what's 0:32:36.169 sometimes called a wireless router. 0:32:38.161 That's the thing that transfers data from the air onto a wire 0:32:41.677 network. You may have a box from your 0:32:43.787 cable company that transfers data from your home out over the 0:32:47.303 cable. And then the cable company 0:32:49.178 itself has a box that transfers data to some other service 0:32:52.518 provider, which may be transfers data across the country, 0:32:55.799 which transfers data into the home of whoever you're trying to 0:32:59.374 send, or the business of whoever you're trying to send a message 0:33:03.066 to. So you have this set of what we 0:33:07.122 are calling switches here that connect all these various hosts 0:33:11.297 together. And these hosts clearly, 0:33:13.556 there are connections here that are sharing data. 0:33:16.842 So if I look at this link here, if both A and B are trying to 0:33:20.949 send data to C, clearly there's going to be 0:33:23.824 some data for both A and B traveling on this wire. 0:33:28 And so, what we end up with is, sorry, so there is a set of 0:33:32.408 issues that are brought up by the fact that we are sharing 0:33:36.741 this data, and that we have this kind of switched infrastructure 0:33:41.529 which we are running data around in. 0:33:44.19 So the first issue that we need to talk about that obviously 0:33:48.674 comes up is this notion of routing. 0:33:51.258 OK, so when you see a graph like this and you ask the 0:33:55.211 question, well, OK, how does A get its data to 0:33:58.631 C? We have to pick some route 0:34:01.524 which A is going to take along this path, through this network 0:34:05.099 in order to get to C. And, there are a couple of 0:34:07.855 different alternatives, right? 0:34:09.555 So, any time you're sending a message, there has to be 0:34:12.662 something that plans out how that route is going to be 0:34:15.768 transmitted. And commonly what happens is 0:34:18.113 that each one of these intermediate points along the 0:34:21.103 network has some table which says how to get to every other 0:34:24.503 location within the network. And we'll talk about how those 0:34:27.903 tables work, how these tables that talk about where to forward 0:34:31.479 your messages to work. But when a node has multiple 0:34:36.034 sort of possible locations where it can send data out to, 0:34:40.102 typically that node is called a router. 0:34:42.863 OK, so if you hear the term router, just think of it as one 0:34:47.076 of these switch things that gets to make a decision about where 0:34:51.581 some kind of traffic is going to be sent to. 0:34:54.705 OK, so now what we need to do in order to think a little bit 0:34:58.991 more about how these switches work is to resolve the issue of 0:35:03.35 how we multiplex traffic over one of these links. 0:35:08 So we said there may be traffic for both A and B running on this 0:35:13.225 link going into D. Well, so how are we going to 0:35:17.041 interleave the traffic from A and B in order to make this 0:35:21.686 work? And there are two techniques. 0:35:24.506 We call this technique for interleaving

data, 0:35:28.156 we call it switching or multiplexing. 0:35:31.142 And we're going to talk about two techniques for switching. 0:35:35.953 We're going to talk about circuit switching. 0:35:39.52 And we'll talk about packet switching. 0:35:44 So circuit switching is pretty easy to understand at an 0:35:47.318 intuitive level. The idea with the circuit 0:35:49.837 switch is that we want to set up some reserved channel that is 0:35:53.586 the communication channel between two endpoints, 0:35:56.474 say for example, A and C. 0:35:57.949 So, a simple way if you want to think about circuit switching is 0:36:01.821 to think about the way that phone networks used to work. 0:36:06 So you may have seen these old movies where the operator would 0:36:09.668 plug wires into a switchboard. What an operator is doing there 0:36:13.336 is establishing a physical circuit, a physical wire that 0:36:16.643 connects two endpoints together. OK, so that's circuit 0:36:19.83 switching. Each operator has a bunch of 0:36:22.115 lines coming in and a bunch of lines coming out. 0:36:24.942 Somebody connects to the operator and says I want to talk 0:36:28.309 to line A. And the operator establishes 0:36:30.594 the physical connection. So, of course we don't use that 0:36:35.532 technology for doing circuit switching anymore, 0:36:39.29 but this notion of circuit switching is still used in the 0:36:43.864 telephone system. And the way that a common 0:36:47.295 digital telephone system might work is as follows. 0:36:51.298 So the idea is that suppose I have some set of nodes A, 0:36:55.709 B, and C, all of which want to send their data out over some 0:37:00.529 line. So they are connecting up to 0:37:03.847 some switch. And they want send their data out 0:37:06.684 over this switch, out over this wire that is on 0:37:09.718 the other side of this switch. OK, so what this kind of 0:37:13.281 switching that we are going to talk about does, 0:37:16.315 which is commonly called TDM for Time Division Multiplexing 0:37:20.142 -- 0:37:21 0:37:31 -- is to give each one of these nodes, A, B, and C, 0:37:34.179 a little unit of time in which only its traffic is being 0:37:37.676 transmitted on this wire. So if you think of this wire as 0:37:41.236 having one additional piece of information put on at each unit 0:37:45.115 of time, and this is the wire stretching out away from the 0:37:48.739 host so that this is sort of time is going this way as the 0:37:52.364 signal propagates down the wire. And if you were to look at all 0:37:56.306 the different messages that are on the wire, what you see'll is 0:38:00.057 sort of a repeating pattern of time intervals. 0:38:04 And each of these time intervals would be carved up 0:38:07.896 into a number of slots. OK, and each slot is allocated 0:38:12.025 to exactly one of these nodes. And each node gets the same 0:38:16.467 slot and each one of these larger intervals. 0:38:19.818 OK, so you'd see a pattern like A's traffic, B's traffic, 0:38:24.181 C's traffic, and so on. 0:38:25.896 And then in the next interval we would have A's traffic, 0:38:30.181 B's traffic, and C's traffic again. 0:38:34 So there would be this repeating pattern where each 0:38:37.562 node is associated to the same slot in every one of these 0:38:41.552 intervals, OK? And we call the data that, 0:38:44.402 say, for example, A puts into a slot a frame. 0:38:47.536 The frame is sort of the set of data that is allocated to a 0:38:51.669 node. And, so this is basically the 0:38:54.091 way that circuit switching works. 0:38:56.371 And it's almost exactly the way that the phone network works 0:39:00.575 now. So a phone network carves up 0:39:03.922 the wire into a bunch of these little slots where each 0:39:07.561 conversation is allocated one of these sort of frames in each 0:39:11.68 interval. So in a typical phone network, 0:39:14.358 a frame might be eight bits. And there might be these 0:39:17.928 intervals. There might be 8,000 of them 0:39:20.537 per second. OK, so we have 8,000 intervals 0:39:23.352 per second. 0:39:25 0:39:30 OK, and so what that means is that if each frame is eight bits 0:39:33.953 and there is 8,000 intervals, each sender can send 64,000 0:39:37.583 bits per second. And, it turns out that 64,000 0:39:40.5 bits per second is about enough to be able to encode voice at a 0:39:44.518 sort of acceptable level for people. 0:39:46.787 So we can talk about the capacity of this channel by 0:39:50.092 basically looking at the number of frames that are in one 0:39:53.722 interval. So if we say there are N max 0:39:56.12 frames in one interval, then that specifies the number 0:39:59.555 of conversations that we can simultaneously maintain on this 0:40:03.379 wire. OK, so the behavior of one of 0:40:06.734 these circuit-switched systems is as follows: 0:40:09.364 everybody who's having a conversation has bandwidth 0:40:12.354 allocated to them, and has a frame allocated to 0:40:15.105 them in every interval. So, up to N max conversations 0:40:18.214 can be held simultaneously and there's no problem. 0:40:21.144 But as soon as we get to the N max plus first guy tries to 0:40:24.552 initiate a call, the network will simply refuse 0:40:27.303 to accept that connection because there's no slot 0:40:30.173 available for this guy to do his communication. 0:40:34 And the last little detail about how these circuit switches 0:40:37.751 work is how do we actually establish a connection in the 0:40:41.308 first place? And we need some protocol for 0:40:43.96 actually reserving to a slot for a particular connection between 0:40:48.034 a pair of endpoints. And so typically what you will 0:40:51.268 do is reserve some frame or set of frames to carry information 0:40:55.213 that's used to configure the connection between the 0:40:58.447 endpoints. So, A can request that an 0:41:02.047 interval be assigned to its connection between C, 0:41:05.979 and then it will be by communicating during a special 0:41:10.238 frame. OK, so that's circuit 0:41:12.45 switching. But, circuit switching has a 0:41:15.563 problem. And we're going to need to 0:41:18.348 introduce this notion of packet switching to deal with it. 0:41:24 0:41:28 OK, so the reason that circuit switching has a problem is as 0:41:31.343 follows: because although circuit switching works great 0:41:34.403 for voice where every conversation is essentially a 0:41:37.236 continuous stream of audio that can always be encoded at about 0:41:40.693 64 kb per second, Internet traffic just doesn't 0:41:43.3 work like that, right? 0:41:44.489 People who are communicating on the Internet have very 0:41:47.719 variable demands for the amount of traffic that they want to 0:41:51.063 send. And even taken as an aggregate, 0:41:53.103 taking all the people who are, say, for example at the 0:41:56.106 community of MIT over time, there still is a lot of 0:41:58.94 variation in the network demand. So this is just a plot over the 0:42:04.062 course of the day with samples taken every five minutes of the 0:42:08.122 amount of traffic that was being sent out of one of MIT's labs. 0:42:12.248 OK, and so you don't need to worry too much about what these 0:42:16.174 different lines show. But the point is that there are 0:42:19.634 these periodic spikes. And these spikes represent 0:42:22.829 spikes in the traffic demand of different machines on the 0:42:26.223 network. So you can see that at any 0:42:28.485 given point in time, there can be anywhere from, 0:42:31.613 at its lowest, these numbers are about 40 Mb 0:42:34.474 per second to numbers as high as 100 Mb a second. 0:42:39 So there's a lot of variation in the

demand of traffic at MIT's network. 0:42:42.349 Here's another way of looking at the same traffic. 0:42:45.084 This is an interesting graph. This is showing the amount of 0:42:48.322 traffic that different protocols use. 0:42:50.332 This is a real graph by the way. 0:42:52.063 This is showing the amount of traffic that is transmitted over 0:42:55.468 MIT's network for different types of protocols. 0:42:58.036 And what you see here is that it's really pretty amazing. 0:43:02 You see that Kazaa, this file trading network, 0:43:04.536 is using up a vast majority of the traffic that we send out of 0:43:07.974 MIT. But besides that, 0:43:09.158 there's another interesting point which is that noticed 0:43:12.201 during the months of October and November, the two leftmost lines 0:43:15.809 here, we are sending a lot more traffic than during the month of 0:43:19.36 December. So, if you were at MIT and are 0:43:21.558 trying to plan how much traffic you expected to be transmitted 0:43:24.996 during a month, this makes it hard because 0:43:27.307 there was a 50% drop in the amount of traffic from one month 0:43:30.632 on this Kazaa network. So we need to introduce this 0:43:35.133 notion of packet switching to deal with this problem of 0:43:39.247 unpredictable rates. 0:43:41 0:43:47 OK, so in many kinds of data networks, it's simply not 0:43:50.697 possible to predict exactly how much traffic you're going to 0:43:54.813 need to send at any point in time. 0:43:57.116 And so the idea is to use something called asynchronous 0:44:00.883 multiplexing instead of time division multiplexing. 0:44:05 0:44:09 And the idea is as follows. Suppose we have our three 0:44:13 nodes, A, B, and C all transmitting data through some 0:44:17 switch out over a wire just like we did before. 0:44:20.538 If you look at the traffic that's being transmitted on this 0:44:25 wire, what you'll see is that it's not necessarily going to 0:44:29.461 have the same regular pattern that we had before. 0:44:34 The idea is that any node is allowed to try and start talking 0:44:38.38 on this wire any time that it wants. 0:44:40.935 It's asynchronous. It's not timed. 0:44:43.345 There's no regular slots when nodes are required to transmit. 0:44:47.725 So this might be traffic for A. This might be traffic for A. 0:44:52.033 This might be traffic for B, A, A, and C. 0:44:54.953 OK, so what this suggests, and noticed that these packages 0:44:59.115 of data that are being transmitted are of different 0:45:02.765 sizes. Typically these are called 0:45:05.101 packets. And that's why this is packet 0:45:08.461 switching because we are sending these little packages of information 0:45:11.58 that are of a variable size as opposed to these fixed size 0:45:14.357 frames that we were transmitting before. 0:45:16.257 And notice that there is no requirement that A transmitted 0:45:19.034 any specific time. A is allowed to try and 0:45:21.032 transmit as much as it wants. But this does introduce a 0:45:23.663 complexity because it makes it hard to plan for the capacity of 0:45:26.684 the network. So suppose that each of A, 0:45:29.581 B, and C have a 1 Mb per second link, and suppose that this link 0:45:33.412 is also 1 Mb per second. Now, clearly if A, 0:45:35.966 B, and C are rarely sending any traffic, then everybody may be 0:45:39.675 able to get all its data onto the network. 0:45:42.168 But if A, B, and C all decide that they want 0:45:44.783 to send at the same time at 1 Mb per second, then there's going 0:45:48.554 to be a problem because they're going to be trying to send more 0:45:52.324 data over this link them the link has capacity for. 0:45:55.364 And so what that immediately suggests is that in these boxes, 0:45:59.013 in these packet switch networks, there is some kind of 0:46:02.236 a queue. OK, so we're doing queuing 0:46:06.26 within the network. But it also, 0:46:08.956 so you might ask the question, why don't network designers 0:46:13.913 design these packet switch networks so that you never have 0:46:18.869 a link that's connecting multiple endpoints together that 0:46:23.739 doesn't have sufficient capacity to transmit all of the 0:46:28.434 information from all of the endpoints that may be 0:46:32.608 transmitting data over it? But if you think about this for 0:46:37.65 a second, that's going to be a really bad idea. 0:46:40.099 So suppose we are on the Internet. 0:46:41.856 And suppose that I am designing some little website. 0:46:44.306 I want to set up a website for students of 6.033. 0:46:46.861 If I needed to make sure that that website had sufficient 0:46:49.843 bandwidth that could support connections from everybody who 0:46:52.931 is on the Internet just in case everybody who's on the Internet 0:46:56.232 decided to access that website at exactly the same time, 0:46:59.161 I would be in big trouble, right, because I would have to 0:47:02.143 have hundreds of terabits of bandwidth available to me in 0:47:05.124 order to be able to support that. 0:47:08 So clearly I'm not going to do that. 0:47:10 Instead, what I'm going to do as a network designer is simply 0:47:13.428 try and ensure that most of the time my network has enough 0:47:16.685 bandwidth available for the users were going to be accessing 0:47:20.057 it. But some of the time, 0:47:21.428 there may not be. I may sort of under-provision 0:47:24.057 the network, may not provide enough resources. 0:47:26.628 And so during those times when the network has more data that's 0:47:30.171 trying to be sent over it than it can possibly send, 0:47:33.199 we call those congestion times. And when a network is subject 0:47:37.686 to long periods of congestion, these queues that are sitting 0:47:41.116 in the network may fill up. So these queues are some fixed 0:47:44.43 size. They are stored typically in 0:47:46.348 RAM on a device, and they can over-fill. 0:47:48.616 And when they over-fill, we have to do something about 0:47:51.697 it. OK, so this problem of 0:47:53.151 congestion in networks leads to all sorts of kind of interesting 0:47:56.813 trade-offs in the design of these packet switch networks. 0:48:01 And typically the bottom line is basically, 0:48:03.577 and we'll talk about this in much more detail, 0:48:06.339 but the bottom line is typically these packet switch 0:48:09.469 networks have to be able to drop data some of the time. 0:48:12.783 If a queue over-fills, the network has no choice but 0:48:15.913 to refuse to transmit data for somebody. 0:48:18.306 But because these packet switch networks are composed of 0:48:21.682 multiple routers along the way, it may be that the data is 0:48:25.18 actually dropped deep in the network by some router not when 0:48:28.801 I first try and send the data, but five hops down the line the 0:48:32.545 data may be dropped. And so these kinds of things 0:48:36.991 complicate the design of these systems. 0:48:39.693 Queuing means there's variable delay. 0:48:42.253 OK, and congestion means there may be data that's lost. 0:48:46.093 And to deal with lost, we may have to retransmit 0:48:49.435 packets. And so this gets the sort of 0:48:51.995 fundamental notion that's going to come up in the design of 0:48:56.119 these packet switch networks, and that's that the extraction 0:49:00.315 that these packet switch networks provides is so-called 0:49:04.155 best effort networking. And that means at the lowest 0:49:08.37 level, these networks don't guarantee that the data that you 0:49:11.259 send over them will actually be delivered. 0:49:13.267 All they say is the network is sort of promising that it will 0:49:16.205 try hard to

get the data to be delivered. 0:49:18.163 But you as a user, the applications that are using 0:49:20.563 this raw network interface need to understand that there is variable 0:49:23.647 delay, they need to understand that there's congestion that can 0:49:26.683 happen which means that packets can be dropped. 0:49:28.935 Congestion also means that typically packets may be 0:49:31.384 retransmitted. And so that means that all of 0:49:34.735 your data may not arrive at the other end. 0:49:37.188 And so this is the best effort network abstraction, 0:49:40.179 and we're going to talk about these kinds of best effort 0:49:43.47 network exclusively for the next several weeks as 0:49:47.119 opposed to these circuit switch networks. 0:49:49.512 And what we're going to do is we're going to see how we can 0:49:52.982 build up a set of software that runs on top of these best effort 0:49:56.752 networks that allows us to provide things like, 0:49:59.504 for example, some reasonable guarantee of 0:50:01.897 reliable delivery, that data won't be lost as data 0:50:04.829 is transmitted over the network. So we will see how we build up 0:50:09.797 layers on top of this best effort network stack that allow 0:50:13.213 us to provide certain guarantees to the user. 0:50:15.849 So that's it for today, and tomorrow we will talk 0:50:18.725 about, starting Monday we will talk about the link layer 0:50:22.021 interface for these things. Don't forget about the exam.